CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

CRT-799

An Optimal Algorithm for the Mixed Chinese Postman Problem*

by
Yves Nobert[1,2]
and
Jean-Claude Picard[2]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, Succursale A, Montréal, Canada H3C 3J7.
[2] Département des sciences administratives, Université du Québec à Montréal, 1495 rue Saint-Denis, Montréal, Canada H3C 3P8.

* This work was supported by the Canadian Natural Sciences and Engineering Research Council (Grants OGP0005486 and OGP0005811).

Centre de recherche sur les transports — Publication #799
October 1991

2-007137

Dépôt légal – 4ème trimestre 1991

ABSTRACT

The Chinese Postman Problem is well solved when the original graph contains only arcs or only edges. The mixed Chinese Postman Problem (MCPP) is however NP-complete and very few papers have been devoted to this problem. In this paper, we present a new integer programming model and a new optimal algorithm for the MCPP.

The simplex method is used iteratively to obtain sharp lower bounds by solving successive relaxations of this model. Optimality is achieved by using Gomory cuts, blossom inequalities and balanced set constraints. Elaborate computer results are presented.

Keywords: Chinese, postman, problem, mixed, simplex, Gomory cuts, blossom inequalities, minimal cuts, 0–1 quadratic programming.

RÉSUMÉ

Le problème du postier chinois peut être résolu par un algorithme polynômial lorsque le graphe original ne contient que des arcs ou des arêtes. Le problème du postier chinois mixte (PPCM) est toutefois NP-complet et peu d'articles ont traité de ce problème. Nous présentons dans cet article un nouveau modèle et un nouvel algorithme pour résoudre le PPCM.

La méthode du simplexe est utilisée pour résoudre une suite de relaxations linéaires de plus en plus forte du modèle en nombres entiers. L'utilisation de coupes de Gomory, de contraintes d'équilibre et de contraintes de parité permettent de trouver rapidement une solution optimale. Des résultats détaillés de tests sur ordinateur sont aussi présentés et commentés.

Mots-clés: Problème, postier, Chinois, mixte, simplexe, coupes de Gomory, coupes, minimales, programmation, quadratique.

1. Introduction

In this paper we consider the mixed Chinese Postman Problem abbreviated in the following by MCPP. Let *G=(N,A,E)* be a graph where *N*={1,...,*n*} is a set of nodes, $A = \{(i,j) : (i,j) \epsilon N \times N, i \neq j\}$ is a set of arcs and $E = \{(i,j) : (i,j) \epsilon N \times N, i < j\}$ is a set of edges. The arcs or edges of G will be termed the links of the graph. We suppose known non-negative lengths associated with each link of G. More precisely, c_{ij} and $d_{k\ell}$ will represent respectively the length of arc *(i,j)* in A and the length of edge (k,ℓ) in E.

The graph G is said to be unicursal or Eulerian if there exists a closed circuit passing through each link of G exactly once in a continuous manner while respecting the orientation of each arc. The MCPP is the problem of adding copies of links of G in order to render the resulting graph Eulerian while minimizing the total length of the links added. If $A = \emptyset$ then the MCPP reduces to the Chinese Postman Problem (CPP) which can be solved, as shown by Edmonds and Johnson [4], as a minimal weighted 1–matching between the odd nodes of G. In the case where $E=\emptyset$, the MCPP is known as the Directed Chinese Postman Problem (DCPP) and is solved by finding a minimum cost circulation [5].

The MCPP is however NP-complete [13] and very few papers have been devoted to this problem. Frederickson et al. [6] as well as Edmonds and Johnson [4] have constructed approximate algorithms for the MCPP. Kapauf and Koehler [9] have studied the mathematical properties of a mathematical model of the MCPP and suggested some ideas for solving this problem exactly. Minieka [11] also proposed

an optimal approach to the MCPP. However, no computational results are reported in these last two papers.

To our knowledge, apart from our algorithm, only two other optimal algorithms have been developed for the MCPP. Christofides et al. [3] used a branch-and-bound algorithm coupled with Lagrangean relaxation while Grötschel and Win [7] devised a cutting planes method for the windy postman problem (WPP) and applied it to the MCPP, which can be viewed as a special case of the WPP. We will comment later on the performance of these two methods. Note finally that an MCPP with m links can be reduced into an equivalent Traveling Salesman Problem (TSP) with $6m+1$ nodes. To do so, one must combine network-transformation techniques elaborated on in [1,8,14].

The MCPP is important in practice and closely related to many problems encountered in municipal management such as snow removal, garbage collection, street sweeping, etc. (see [2]). A good algorithm for the MCPP, in terms of the quality of the solutions obtained, can thus lead to significant cost savings for numerous real arc-routing problems.

2. Notation

The following notation will be used throughout this paper.

- S is a subset of N.
- $\overline{S} = N - S$.

- $A^+(S) = \{(i,j) \epsilon A : i \epsilon S, j \epsilon \overline{S}\}$ is the set of arcs leaving S.
- $D^+(S) = |A^+(S)|$.
- $A^-(S) = \{(i,j) \epsilon A : i \epsilon \overline{S}, j \epsilon S\}$ is the set of arcs entering S.
- $D^-(S) = |A^-(S)|$.
- $E(S) = \{(i,j) \epsilon E : (i \epsilon S, j \epsilon \overline{S})$ or $(i \epsilon \overline{S}, j \epsilon S)\}$ is the set of edges between S and $\overline{S}$.
- $D(S) = |E(S)|$.
- $A(S) = A^+(S) \cup A^-(S)$ is the set of arcs leaving or entering S.
- When $S = \{i\}$ the preceding notations will be written $A_i^+, D_i^+, A_i^-, D_i^-, E_i, D_i$ and A_i.
- $d_i = D_i^+ + D_i^- + D_i$ is the degree of node i.

3. Mathematical formulation of the MCPP

Ford and Fulkerson [5] have shown that a mixed graph is Eulerian if and only if

a) the graph is strongly connected;

b) the degree of each node is even **(evenness condition)**;

c) for all proper subsets S of nodes, the number of arcs leaving S minus the number of arcs entering S is less than or equal to the number of edges between S and its complement **(balanced sets condition)**.

In the following we suppose that G satisfies condition a) otherwise the MCPP is infeasible. A graph for which condition b) (c)) is satisfied is said to be even (balanced).

Let x_{ij} and y_{ij} represent respectively the optimal numbers of copies of arc $(i,j) \epsilon A$ and edge $(i,j) \epsilon E$ which, added to the graph G, make it Eulerian. Define the graph

$G_{x,y}$ as the original graph with the additional copies of links indicated by the values of x_{ij} and y_{ij}. More precisely, if $x_{ij} = p$ then $G_{x,y}$ will contain $(p+1)$ arcs (i,j). A similar interpretation is given to $y_{ij} = p$.

The graph $G_{x,y}$ whose total length is minimum can be found by solving the following mathematical model.

$$\text{Min} \qquad Z = \sum_{(i,j)\epsilon A} c_{ij}\, x_{ij} + \sum_{(i,j)\epsilon E} d_{ij}\, y_{ij} \tag{1}$$

subject to

$$\sum_{(i,j)\epsilon A_k} x_{ij} + \sum_{(i,j)\epsilon E_k} y_{ij} = 2v_k + p_k \qquad (k\epsilon N) \tag{2}$$

$$(P) \qquad -\sum_{(i,j)\epsilon A^+(S)} x_{ij} + \sum_{(i,j)\epsilon A^-(S)} x_{ij} + \sum_{(i,j)\epsilon E(S)} y_{ij} \geq u_S \qquad (S \subset N) \tag{3}$$

$$x_{ij} \geq 0 \text{ and integer } ((i,j)\epsilon A) \tag{4}$$

$$y_{ij} \geq 0 \text{ and integer } ((i,j)\epsilon E) \tag{5}$$

$$v_i \geq 0 \text{ and integer } (i\epsilon N) \tag{6}$$

In (1) we simply state that the total length of the copies added must be minimized in the MCPP. The constraints of (2) relate to the evenness condition. In a mixed graph, the degree of a node is simply the number of links incident to it. Let $p_k = 1$ if the

degree of node k is odd in G (k is called odd) and $p_k = 0$ otherwise (k is called even). The constraints of (2) imply that an odd (even) number of copies of links will be added to an odd (even) node of G, so that each node of $G_{x,y}$ will be even.

Closely related also to the evenness condition are the blossom inequalities defined by Edmonds and Johnson [4] for the CPP, and generalized here in (7) for the MCPP:

$$\sum_{(i,j)\epsilon A(S)} x_{ij} + \sum_{(i,j)\epsilon E(S)} y_{ij} \geq 1 \quad \forall S \text{ odd} \tag{7}$$

where a subset S of nodes is called odd in a graph if it contains an odd number of odd nodes. It is easily seen that S is odd if and only if $D^+(S) + D^-(S) + D(S)$ is also odd. Since, in any Euler tour in $G_{x,y}$, the frontier between S and $\overline{S}$ will be crossed an even number of times, (7) is valid for the MCPP. We denote by $\beta(S)$ the blossom inequality associated with an odd set S. When $S = \{i\}$ we simply write β_i.

The blossom inequalities are redundant relative to (2), (4), (5), (6). We will see however that, since both (2) and the integrality condition on the variables are relaxed initially in our algorithm, the constraints of (7) are very useful for imposing the evenness condition as well as for cutting fractional solutions.

The constraints of (3) assure that $G_{x,y}$ will satisfy the balanced sets condition. Let $u_S = D^+(S) - D^-(S) - D(S)$ for $S \subset N$ (u_i if $S = \{i\}$). If $u_S > 0$ then the number of times one must leave S $\left(D^+(S)\right)$ exceeds by u_S the number of times one can enter S $\left(D^-(S) + D(S)\right)$. Since each link of G must be traversed at least once then the net number of copies of links that must be added in G from $\overline{S}$ to S is at least u_S. Such a set S is said to be outwards unbalanced. It is important to note

here that (3) deals only with outwards unbalanced sets. Since an inwards unbalanced set $S\big(D^-(S) > D^+(S) + D(S)\big)$ has an outwards unbalanced complement, it is then sufficient to consider the constraints of (3) in order to impose the balanced sets condition in $G_{x,y}$.

If $u_S \leq 0$, the set S is not outwards unbalanced. In order to give an interpretation of (3) in this case, we first multiply both sides of (3) by -1 to obtain

$$\sum_{(i,j)\epsilon A^+(S)} x_{ij} - \sum_{(i,j)\epsilon A^-(S)} x_{ij} - \sum_{(i,j)\epsilon E(S)} y_{ij} \leq -u_S = t_S . \qquad (8)$$

When $t = \sum_{(i,j)\epsilon A^-(S)} x_{ij} + \sum_{(i,j)\epsilon E(S)} y_{ij} = 0$ the quantity t_S is an upper bound on the number of copies of links that can be added from S to $\overline{S}$ in order to keep S outwards balanced. If $t > 0$, this upper bound becomes $t_S + t$. The constraints of (3) are called the balanced sets constraints.

Note finally about (3) that when $E(S) = \varnothing$ then (3) can be written as an equation. This is easily seen by considering in this case the particular constraints of (3) associated with S and $\overline{S}$. In the following, the constraints of (3) will be denoted by $u\beta(S)$ for $S \subset N$ and by $u\beta_i$ when $S = \{i\}$. In the rest of the paper, sets T such that

$$u_T = \max_{S \subset N} \{u_S\}$$

are called **most unbalanced sets**. These sets play a crucial role in our algorithm.

For illustration consider the graph depicted in figure 1:

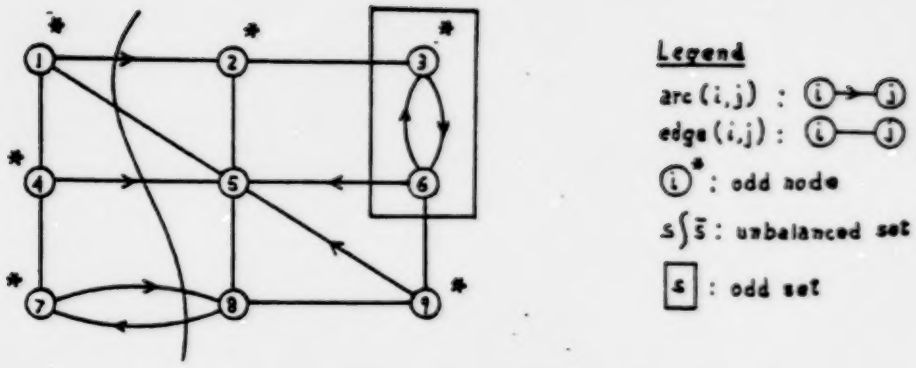

Figure 1: A mixed graph and some constraints of P

The following constraints constitute some blossom inequalities and balanced sets constraints that we can derive for this graph:

$$x_{12} + y_{14} + y_{15} \geq 1 \quad (\beta_1)$$

$$x_{65} + y_{23} + y_{69} \geq 1 \quad (\beta(3,6))$$

$$x_{87} + y_{15} - x_{12} - x_{45} - x_{78} \geq 1 \quad (u\beta(1,4,7))$$

$$x_{65} - y_{23} - y_{69} \leq 1 \quad (u\beta(3,6)).$$

If the original graph is Eulerian then the MCPP is solved trivially with no copies of links being added in G. In this case, $p_k = 0$ for all nodes k, there are no odd sets and $u_S \leq 0$ for all $S \subset N$. Hence letting all the variables equal 0 is clearly feasible and also optimal for P since all the lengths of the links are non-negative.

The model P contains an exponential number of constraints whereas most models encountered in the literature [3,7] use a polynomial number of constraints to define the feasible solutions of the MCPP. The main difficulty in these models is achieving integrality (see Section 5).

4. An Optimal Algorithm for the MCPP

4.1 General description of the algorithm

Due to the large number of constraints in P, it is impossible, even for small n, to solve this problem directly by writing down explicitly all the constraints and using a branch-and-bound or a cutting planes code. Instead, we will solve a sequence of linear programs containing few but effective constraints. We prefer in this section to present the algorithm in a general framework, details pertinent to the main steps of the algorithm being the subject of independent sections. We suppose for the time being that it is possible to detect in an efficient way odd or unbalanced sets and so, to determine if a given graph satisfies the evenness and balanced sets conditions.

We need the following notation:

P_k is the relaxation of P at iteration k of the algorithm;

$v(P_k)$ denotes the optimal value of problem P_k;

m_k is the number of constraints in P_k (excluding non-negativity constraints);

(x^k, y^k) constitutes an optimal solution of P_k.

The algorithm goes as follows:

Step 0. (Initial phase: construction of the initial relaxation).

- Construct P_0 as the linear program in variables x, y with the following form:

  Min Objective function (1)

  subject to:

- blossom inequalities (7) associated with the odd nodes of G;
- balanced sets constraints (3) associated with the unbalanced nodes and most unbalanced sets of G;
- non-negativity on the variables.

- Set m_0 equal to the number of constraints in P_0 excluding the non-negativity constraints and set $k = 0$
- If $m_0 = 0$ then G is Eulerian and go to Step 5.

Step 1. (Simplex phase).

- Evaluate (x^k, y^k) and $v(P_k)$ by solving P_k with the simplex method.

Step 2. (Restriction phase: balanced sets condition).

- If G_{x^k,y^k} is balanced then go to Step 3.
- Append explicitly to P_k balanced sets constraints that are violated by (x^k, y^k).
- Let p_1 be the number of constraints added.

Step 3. (Restriction phase: Evenness condition).

- Try to add to P_k blossom inequalities that are violated by (x^k, y^k) and that are not implied by the constraints generated at Step 2.
- Let p_2 be the number of blossom inequalities newly inserted in P_k.
- If $p = p_1 + p_2 = 0$ then go to Step 4. Otherwise:
  - set $k = k + 1$ and $m_k = m_{k-1} + p$
  - go to Step 1.

Step 4. (Restriction phase: Integrality on the variables).

- If (x^k, y^k) is not integral then
  - add one Gomory cut to P_k
  - set $k = k + 1$ and $m_k = m_{k-1} + 1$
  - go to Step 1.
- If there is an odd node i in G_{x^k,y^k} then
  - generate the appropriate constraint of (2) and the corresponding v_i variable
  - add a Gomory cut on v_i to P_k
  - set $k = k + 1$ and $m_k = m_{k-1} + 1$
  - go to Step 1.

Step 5. (Identification of a Euler tour).

- Set $v(P) = v(P_k)$.
- Determine a Euler tour in G_{x^k,y^k}.
- End.

The previous algorithm is a dual method where the first feasible solution is optimal. In general its efficiency mainly depends on two factors. Firstly, we must be able to solve rapidly the different relaxed problems encountered in Step 1. If the total number of constraints generated in the process stays reasonably small, the simplex method will perform in an efficient way. Secondly, the binding constraints defining the optimal extreme point of the original problem must be identified early in the process after few iterations. If this should not happen, the number of constraints

generated will grow large and hence much time will be spent in simplex iterations, to say nothing of space requirements which can become problematic. Very often in this situation, numerical instability sets in and the whole process does not converge. A good idea for avoiding this situation is to always generate strongly violated constraints. This should ensure that $v(P_k)$ will converge to $v(P)$ rapidly.

The efficiency of this relaxation process has been illustrated by many authors on numerous combinatorial optimization problems. In particular, Nobert [12] has applied successfully a similar method to the symmetrical TSP and some of its straightforward generalizations.

Another important feature of the algorithm presented above is that some of the variables (the v_i's) are also relaxed. Including these variables in the initial step of the algorithm does not affect the main ideas of the method nor its efficiency in terms of time. The results given later in section 5 of many computational tests will confirm this assertion.

In order that the algorithm for exact solution of the MCPP given in this section be implemented, we must be able to circumvent two difficulties, namely:

1. How to identify unbalanced sets.
2. How to identify violated blossom inequalities.

These two difficulties are not trivial at all to overcome. The way we achieve this can be considered as one of the main scientific results of this paper. Moreover, the order in which the steps of the algorithm are executed has been very carefully established. There exist many other possibilities and we have presented here only the

order that has given the best computational results. Finally, note that the particular blossom or balanced sets constraints that are appended to the model have been chosen with caution in order to avoid redundancy and to ensure that only the strongest ones are selected. We explain all this in greater detail in the following subsections.

4.2 Balanced sets condition, Steps 0 and 2 of the algorithm

The ability to determine if a given graph is balanced and if not, to detect unbalanced sets is crucial in Step 0 and Step 2 of the algorithm presented in 4.1. One should understand that by choosing variables (the y_{ij}'s) that do not indicate the direction in which the edges of G_{x^k,y^k} will be traversed, as is usually the case in the literature, the task of verifying the balanced sets condition is a complicated one. For example, let us consider the graph depicted in Figure 2.

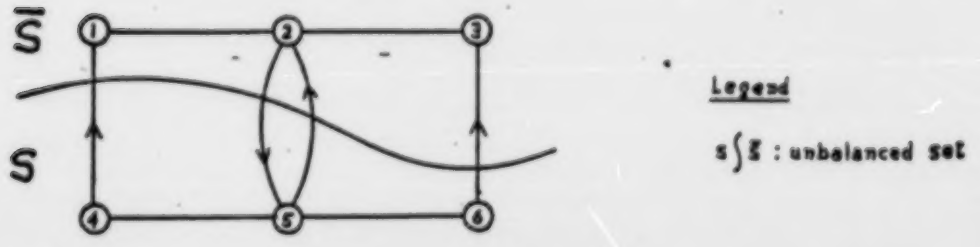

Figure 2: An even unbalanced graph

All the nodes of this strongly connected graph are even and balanced. However, this graph is unbalanced since S={4,5,6} is outwards unbalanced by 2. When a graph is completely directed or when one knows the directions of the edges by the

choice of variables, then the balanced sets condition and the evenness condition are essentially the same. In this section, we devise a polynomial method for finding one most unbalanced set in a graph, enabling us to check if the graph is balanced.

Let $\widetilde{G} = \left(\widetilde{N}, \widetilde{A}, \widetilde{E}\right)$ be an arbitrary mixed multigraph with n nodes where $\tilde{a}_{ij}$ and $\tilde{u}_{k\ell}$ respectively represent the number of arcs of the form (i,j) in $\widetilde{A}$ and the number of edges of the form (k,ℓ) in $\widetilde{E}$. We associate with this graph a quadratic function $f_{\widetilde{G}}(x_1,...,x_n)$ defined as follows:

$$f_{\widetilde{G}}(x_1,...,x_n) = \sum_{(i,j)\in\widetilde{A}} \tilde{a}_{ij}\, x_i(1-x_j) - \sum_{(i,j)\in\widetilde{A}} \tilde{a}_{ij}\, x_j(1-x_i) - \sum_{(i,j)\in E} \tilde{u}_{ij}\,(x_i - x_j)^2 \tag{9}$$

We denote by $f_{\widetilde{G}}(S)$ the value taken by $f_{\widetilde{G}}(x_1,...,x_n)$ when $x = (x_1,...,x_n)$ is the characteristic vector of $S \subset \widetilde{N}$, i.e. $x_i = 1$ if $i \in S$ and $x_i = 0$ otherwise. After some manipulations we obtain easily

$$f_{\widetilde{G}}(S) = D^+(S) - D^-(S) - D(S) = u_S \tag{10}$$

where $D^+(S), D^-(S), D(S)$ and u_S are defined as in section 3 but for the graph $\widetilde{G}$. By (10) we see that $f_{\widetilde{G}}(S)$ will take a positive value if and only if S is outwards unbalanced in $\widetilde{G}$. If $S = \emptyset$ then $x_i = 0$ for all i and $f_{\widetilde{G}}(S) = 0$, so that

$$f^*_{\widetilde{G}} = \max_{S\subset\widetilde{N}} f_{\widetilde{G}}(S) \geq 0. \tag{11}$$

We conclude from (11) that $\widetilde{G}$ does not contain any unbalanced set if and only if $f^*_{\widetilde{G}} = 0$. In particular, by letting $\widetilde{G} = G_{s^k,g^k}$, the problem of verifying if a graph

satisfies the balanced sets condition, encountered in Step 0 and Step 2 of our algorithm, reduces to finding the maximum of a quadratic function in binary variables such as (9).

After some manipulations, the quadratic function $f_{\tilde{G}}(x)$ can also be written in the following form:

$$f_{\tilde{G}}(x) = \sum_{i=1}^{n} (d_i^+ - d_i^- - d_i)x_i + \sum_{(i,j)\in\tilde{E}} 2\tilde{u}_{ij}\, x_i\, x_j \qquad (12)$$

Since all quadratic terms $x_i x_j$ have a non-negative coefficient $2\tilde{u}_{ij}$, the maximum value of $f_{\tilde{G}}(x)$ can be found by a polynomial algorithm using the work of Picard and Ratliff [16]. They showed that maximizing such a quadratic function is equivalent to solving a maximum flow problem on an associated graph with $(n + 2)$ nodes. Thus, the problem of deciding if a given mixed multigraph satisfies the balanced sets condition can be solved in an efficient way.

The work of Picard and Queyranne [15] enables us to go one step further. By deriving yet another graph G^* associated with the optimal solution of the maximal flow problem related to $f_{\tilde{G}}(x)$, the problem of finding all the most unbalanced sets reduces to that of finding all the minimum cuts in G^*. In theory, minimum cuts can be exponential in number [15] and so we devised a simple enumeration scheme where the number of most unbalanced sets detected was limited by a predetermined upper bound.

At Step 0 of our algorithm we need to find blossom inequalities and balanced sets constraints associated with each node i of G. Since this essentially amounts to

counting edges incident to i and arcs leaving or entering node i, we can conclude that the initial relaxation P_0 can be constructed in polynomial time. It should be clear also that the same comment also applies to Step 2 of the algorithm. One important feature of our algorithm is that each time the initial graph G or the subsequent graphs G_{x^k,y^k} are unbalanced, at least one violated balanced set constraint will be inserted in the current relaxation. We do not give here the precise balanced set constraint that will be appended to the model at Step 0 or Step 2. We leave that to Section 4.4 where details concerning the implementation of the algorithm are given.

Two final comments are in order here. Firstly, loosely speaking, the role of the constraints in P_0 is to correct violations of the evenness and the balanced sets conditions in the original graph. In particular, the constraints of P_0 will all have a positive right-hand side. In the subsequent steps of the algorithm, a most unbalanced set found in G_{x^k,y^k} at Step 2 can be unbalanced or balanced in G but rendered unbalanced because of the value assumed by (x^k, y^k). In this last case, the constraint of (3) generated can be seen to have a less-than-or-equal sign with a positive right-hand side. In conclusion, what our algorithm actually does in Step 2 is to eliminate some original violations of the balanced sets condition while trying not to unbalance sets that were balanced in G.

Secondly, Step 2 of the algorithm can be entered with (x^k, y^k) fractional. Nevertheless, all the material presented here is still valid. Our tests have clearly shown that it is preferable to cut fractional solutions with balanced sets constraints rather than to use Gomory cuts.

We conclude this section with an example. Consider the graph in Figure 3.

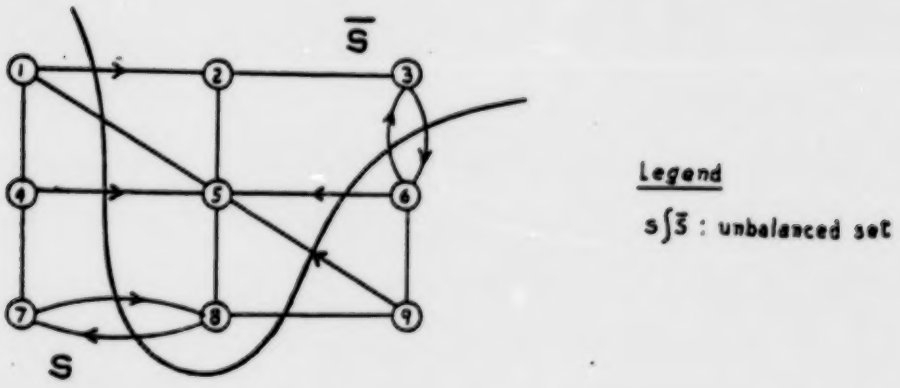

Figure 3: A most unbalanced set

The quadratic function associated with the graph in Figure 3 is:

$$\begin{aligned} f_{\widetilde{G}}(x) = & -x_1 - 3x_2 - x_3 - x_4 - 6x_5 - x_7 - 2x_8 - x_9 \\ & +2x_1x_4 + 2x_1x_5 + 2x_2x_3 + 2x_2x_5 \\ & +2x_4x_7 + 2x_5x_8 + 2x_6x_9 + 2x_8x_9 \end{aligned}$$

The function $f_{\widetilde{G}}(x)$ is maximized when $x = (1,0,0,1,0,1,1,0,1)$ and so $S = (1,4,6,7,9)$ is a most unbalanced set in this graph. The associated balanced set constraint is:

$$x_{87} + x_{36} + y_{15} + y_{89} - x_{12} - x_{45} - x_{78} - x_{95} - x_{65} - x_{63} \geq 2. \qquad (14)$$

Note that $S_1 = (1,4,6,7,8,9)$ is the only other most unbalanced set in this graph. We refer the reader to [15,16] to see how the detailed calculations pertaining to minimum cuts (most unbalanced sets in our case) are done.

4.3 Evenness and integrality conditions: Steps 3 and 4 of our algorithm

In Step 3 of the algorithm we want to eliminate solutions (x^k, y^k) that do not satisfy the evenness condition. One should be reminded that the variables v_i are not present in the initial relaxation P_0 and so, if a node i happens to be odd in G_{x^k,y^k}, it is not possible to calculate a Gomory cut on v_i, which is then evidently fractional, unless v_i and the corresponding constraint of (2) have been or are inserted in the model. From a computational standpoint, it is usually preferable to resort to Gomory cuts only when one has no better choices. One reason is that the density of non-zero coefficients in Gomory cuts is usually quite large. Moreover, the kind of method we have devised with, at its center the simplex method, performs better in terms of time and space requirements with as few variables and constraint coefficients as possible. This line of reasoning has led us to relax all the v_i variables initially and to use blossom inequalities, instead of Gomory cuts, as often as possible, in order to impose the evenness and integrality conditions. We explain in this section how this is achieved.

Let $G^A_{x^k,y^k} = (N, E_{x^k,y^k})$ be a non-oriented graph where E_{x^k,y^k} is given by:

$$E_{x^k,y^k} = \left\{(i,j) : i < j, \ \left(x^k_{ij} > 0 \text{ or } x^k_{ji} > 0 \text{ or } y^k_{ij} > 0\right)\right\}. \qquad (15)$$

There is an edge (i,j) in E_{x^k,y^k} if and only if (i,j) or (j,i) is a link in G and the corresponding variable x^k_{ij}, x^k_{ji} or y^k_{ij} takes a positive (possibly fractional) value in the optimal solution of P_k.

Let $S_1, ..., S_a$ be the connected components of $G^A_{x^k, y^k}$. Clearly, for any ℓ such that S_ℓ is odd in G, the constraint $\beta(S_\ell)$ is violated by x^k, y^k and can be appended to P_k in order to cut off the solution x^k, y^k. If, however, S_ℓ is even for $\ell = 1, ..., a$ then no violated blossom inequalities can be derived directly from the connected components of $G^A_{x^k, y^k}$. Calculating the connected components of $G^A_{x^k, y^k}$ requires $0(p)$ operations where p is the number of variables taking a positive value in (x^k, y^k). Clearly $p \leq |A| + |E|$ but usually $p \ll |A| + |E|$ since the simplex method tends to give positive value to as few variables as possible. So the whole process of generating blossom inequalities in this way is quite efficient.

When the optimal solution of P_k is integer, for any subset S of nodes, we have

$$\sum_{(i,j) \epsilon A(S)} x^k_{ij} + \sum_{(i,j) \epsilon E(S)} y^k_{ij} = p \geq 0 \qquad (16)$$

with p integer. If S is odd in G and if $\beta(S)$ is violated by (x^k, y^k) then clearly $p = 0$ and so $S = S_\ell$ for some ℓ, implying that $\beta(S)$ will be detected by our procedure.

However, if (x^k, y^k) is fractional, we may encounter a situation where none of the odd sets in G constitute a connected component of $G^A_{x^k, y^k}$, so that no violated blossom inequalities are identified. To see this, let us consider a graph $G^A_{x^k, y^k}$ where one of its connected components S_ℓ has the form shown in Figure 4.

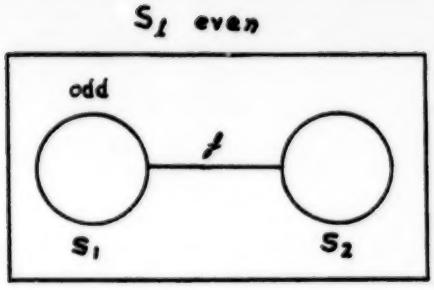

Figure 4: A violated blossom inequality
not detected by our procedure.

Suppose that S_ℓ is even in G and that S_ℓ can be partitioned as depicted in Figure 4 into two sets S_1 and S_2 such that

(i) S_1 is odd in G;

(ii) $\sum_{(i,j)\epsilon A(S_1)} x_{ij}^k + \sum_{(i,j)\epsilon E(S_1)} y_{ij}^k = f$

(iii) $f\epsilon(0,1)$.

Clearly $\beta(S_\ell)$ is not valid since S_ℓ is even in G but $\beta(S_1)$ constitutes a valid blossom inequality that is violated by (x^k, y^k). However, since S_1 is strictly included in a connected component of $G^A_{x^k,y^k}$, $\beta(S_1)$ will not be detected by our procedure.

Hence, there is a major difference between Step 2 and Step 3 of our algorithm. In Step 2, whenever G_{x^k,y^k} is unbalanced, we are assured of finding at least one constraint that cuts (x^k, y^k). In Step 3, it may happen that no violated blossom inequalities are isolated. Step 4 of the algorithm, entered only if the two previous steps did not yield any new constraints, takes care of the integrality condition on

the x_{ij}, y_{ij} and v_i variables with the help of Gomory cuts, the v_i variables being generated if necessary.

4.4 Identifying strong violated constraints and implementation of the algorithm

We now give a detailed description of the particular constraints that are chosen at Step 2, Step 3 and Step 4 of our algorithm. The reader may skip this section without impairing his comprehension of the whole method.

(i) Most unbalanced sets can, in theory, be very numerous even though in practice their number is usually small. Nevertheless, in Step 0, we limited our search to 100. Moreover, from iteration 2 forward, only one most unbalanced set was analyzed. Computational tests showed that balanced sets constraints generated after iteration 0 were mainly not binding at the optimal extreme point.

(ii) A most unbalanced set S is often the union of smaller (in terms of cardinality) disjoint sets $S_1, ..., S_b$ such that, for $1 \leq i < j \leq b$,

$$(A(S_i) \cup E(S_i)) \cap (A(S_j) \cup E(S_j)) = \emptyset.$$

Instead of generating only the constraint $u\beta(S)$, we append to the current relaxation the stronger constraints $u\beta(S_i)$ for each $S_i (1 \leq i \leq b)$ such that $u_{S_i} > 0$. We say that S is "exploded" into $S_1, ..., S_b$. Exploding a set is equivalent to calculating the connected components of the subgraph of G induced by S.

(iii) Clearly $\beta(S)$ is redundant relative to $u\beta(S)$ and so:

- balanced set constraints are generated (Step 2) before blossom inequalities (Step 3).
- every connected component S_i (odd or even in G) of $G^{A}_{x^k,y^k}$ yields a balanced set constraint whenever $u_{S_i} > 0$. Hence $\beta(S_i)$ with S_i odd in G is not inserted in the model if $u_{S_i} > 0$.

(iv) In order to avoid having the total number of constraints become too great, inactive constraints in the optimal solution of P_k are purged from the model for $k \geq 1$. Constraints of P_0 are however essential and hence are never purged. It may happen that a constraint, dropped from the model at an earlier iteration, must be reinserted in the problem later. Computational tests showed however that this situation scarcely occurs and that a great amount of memory was saved by the purge subroutine.

(v) In Step 4, the first basic variable whose value has a positive fractional part is used to generate a Gomory cut. Here again, many tests were done and this proved to be the best among the strategies analyzed.

4.5 Illustration of the algorithm

Figure 5 shows the graph on which the algorithm will be applied.

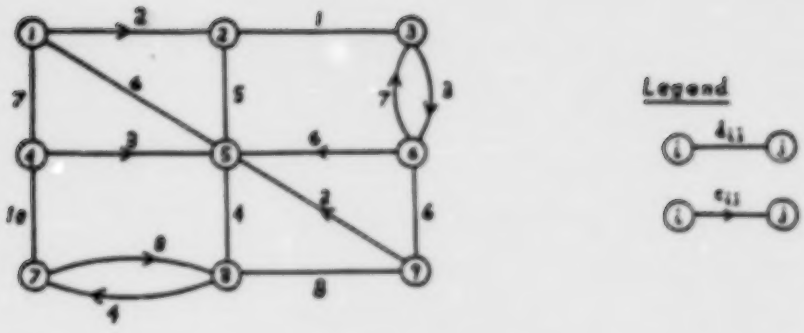

Figure 5: A mixed graph

At each iteration figures will illustrate the graph G_{x^*,y^*} or the graph $G^A_{x^*,y^*}$ with the appropriate information corresponding to the constraints added to the relaxation considered. No constraints were purged in this illustration.

(Initial relaxation: P_0)

Figure 6 illustrates the constraints inserted in the initial relaxation:

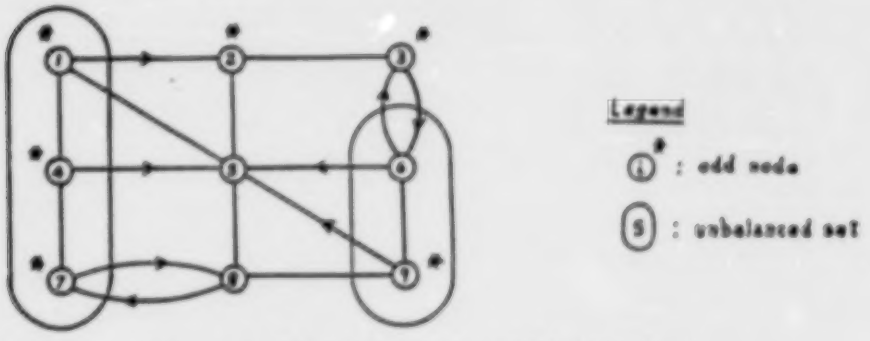

Figure 6: Constraints of the initial relaxation

We notice in Figure 6 that the nodes 1,2,3,4,7 and 9 are odd and so the blossom inequalities $\beta_1, \beta_2, \beta_3, \beta_4, \beta_7$ and β_9 are inserted in P_0. This graph contains two most unbalanced sets, namely $S = \{1,4,6,7,9\}$ and $T = \{1,4,6,7,8,9\}$ with

$u_S = u_T = 2$. The set S is exploded into $S_1 = \{1,4,7\}$ and $S_2 = \{6,9\}$ with $u_{S_1} = u_{S_2} = 1$. The set T cannot be exploded into smaller sets and hence the following balanced set constraints are added in P_0:

$$u\beta(1,4,7),\ u\beta(6,9),\ u\beta(1,4,6,7,8,9).$$

The initial relaxation contains a total of 9 constraints.

(Simplex phase: iteration 0, 9 constraints)

The values of x^0 and y^0 are indicated in Figure 7. The objective function takes the value $v(P_0) = 23$ which is a lower bound on the optimal value of the MCPP.

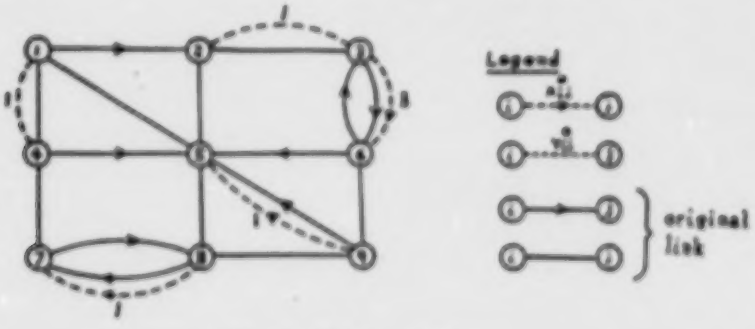

Figure 7: Graph G_{x^0,y^0}

The optimal solution of P_0 is thus

$$x_{36} = 3,\ y_{14} = y_{23} = x_{95} = x_{87} = 1$$

(Restriction phase: balanced sets condition)

By maximizing the quadratic function $f_{G_{x^0,y^0}}$, we find that the set $S = \{3,8,9\}$ is a most unbalanced set with $u_S = 2$. After the explosion procedure the constraints $u\beta_3$ and $u\beta(8,9)$ are added to P_0. This is illustrated in Figure 8.

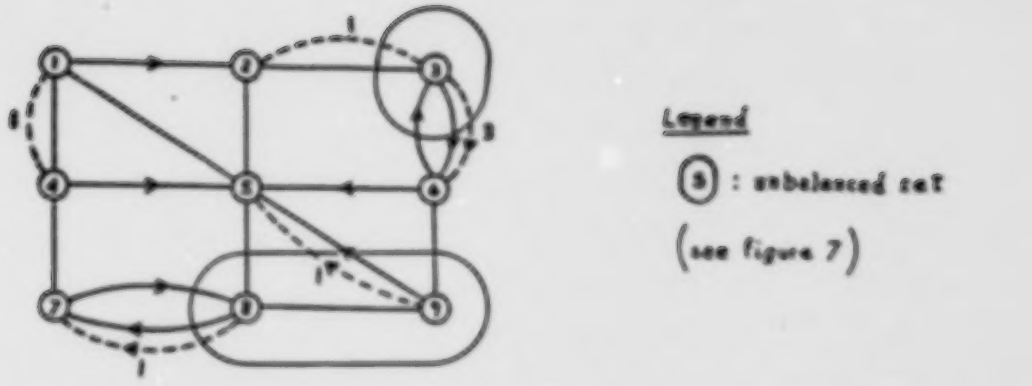

Figure 8: Unbalanced sets in G_{x^0,y^0}

Note that the sets {3} and {8,9} are not balanced in G but are outwards unbalanced by 1 in G_{x^0,y^0}. The constraints $u\beta_3$ and $u\beta(8,9)$ are used here to cut off (x^0, y^0).

(Restriction phase: eveness condition)

The graph $G^A_{x^0,y^0}$ is depicted in Figure 9.

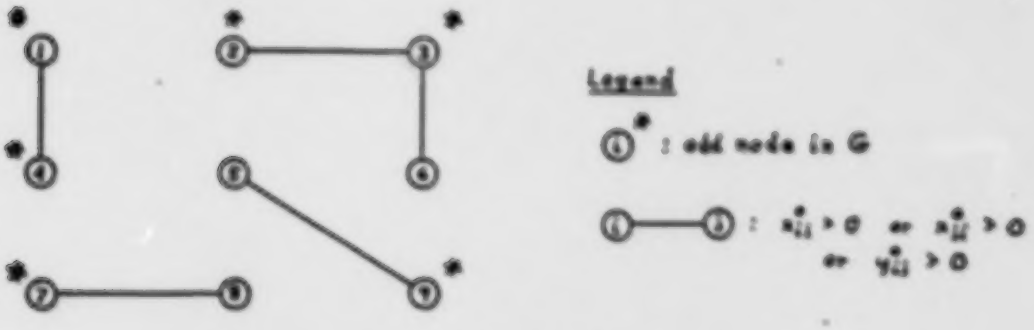

Figure 9: Graph $G^A_{x^0,y^0}$

The graph $G^{A}_{x^1,y^1}$ has four connected components: $S_1 = \{1,4\}$, $S_2 = \{2,3,6\}$, $S_3 = \{5,9\}$ and $S_4 = \{7,8\}$. The sets S_3 and S_4 are odd and the constraints $\beta(5,9)$ and $\beta(7,8)$ are included in P_0.

(Simplex phase: iteration 1, 13 constraints)

Using the simplex method, we obtained the results shown in Figure 10 with $v(P_1) = 24$.

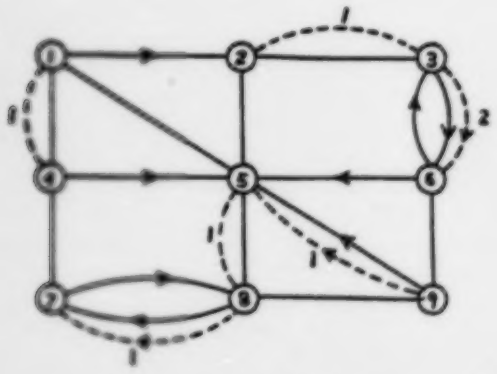

Figure 10: Graph G_{x^1,y^1}

The optimal solution of the relaxed problem is then:

$$x_{36} = 2,\ x_{95} = y_{14} = y_{23} = y_{58} = x_{87} = 1.$$

The restriction phases applied to G_{x^1,y^1} yield no other constraints. In fact one can easily verify that every node in G_{x^1,y^1} is even and that there are no odd connected components in G_{x^1,y^1}. Moreover, $f^{*}_{G_{x^1,y^1}} = 0$ so we can conclude that G_{x^1,y^1} is Eulerian. Since every constraint of P_1 is valid for the MCPP associated with G, we can claim also that G_{x^1,y^1} gives an optimal way of adding copies of links in G in order to make it Eulerian.

It is important to notice here that the complete mathematical model P associated with the graph depicted in Figure 5 contains 9 v_i variables, 16 x_{ij} and y_{ij} variables and a number of constraints proportional to $2^9 = 512$.

We have found the optimal solution of this (quite difficult) integer linear program using only 13 constraints, no v_i variables, no Gomory cuts and a very small number of simplex iterations. One Euler tour in G_{x^1,y^1} is (1–2–3–6–3–6–9–5–2–3–6–5–1–4–5–8–7–8–9–5–8–7–4–1). Intially 16 links were in G and 7 more links were needed to make it Eulerian of minimum length.

We finally give the mathematical model associated with the last relaxation (P_1) obtained by our algorithm in this example:

$$\begin{aligned} \min \quad Z = & 7y_{14} + 10y_{47} + 2x_{12} + 6y_{15} + 3x_{45} \\ & +8x_{78} + 4x_{87} + 2y_{25} + 4y_{58} \\ & +1y_{23} + 6x_{65} + 2x_{95} + 8y_{89} \\ & +7x_{63} + 3x_{36} + 6y_{69} \end{aligned}$$

subject to:

$x_{12} + y_{15} + y_{14} \geq 1$	β_1
$x_{12} + y_{23} + y_{25} \geq 1$	β_2
$y_{23} + x_{63} + x_{36} \geq 1$	β_3
$y_{14} + y_{47} + x_{45} \geq 1$	β_4
$y_{47} + x_{78} + x_{87} \geq 1$	β_7
$y_{69} + x_{95} + y_{89} \geq 1$	β_9
$x_{87} + y_{15} - x_{12} - x_{45} - x_{78} \geq 1$	$u\beta(1,4,7)$
$y_{89} + x_{36} - x_{63} - x_{65} - x_{95} \geq 1$	$u\beta(6,9)$
$y_{15} + y_{58} + x_{36} - x_{12} - x_{45} - x_{95} - x_{65} - x_{63} \geq 2$	$u\beta(1,4,6,7,8,9)$

$$x_{36} - y_{23} - x_{63} \leq 1 \qquad u\beta_3$$

$$x_{87} + x_{95} - x_{78} - y_{58} - y_{69} \leq 1 \qquad u\beta(8,9)$$

$$y_{47} + y_{58} + y_{89} \geq 1 \qquad \beta(7,8)$$

$$y_{25} + x_{65} + y_{69} + y_{89} + y_{58} + x_{45} + y_{15} \geq 1 \qquad \beta(5,9)$$

$$x_{ij} \geq 0 \qquad (i,j) \in A$$

$$y_{ij} \geq 0 \qquad (i,j) \in E$$

5. Computational results and comparison with other methods

Our algorithm was tested on randomly generated graphs. A uniform distribution on (0,1) was used to decide whether a link was to be present between two nodes i and j and to determine its type, i.e. an edge (i,j), an arc (i,j) or an arc (j,i). Two cost structures were studied: a length of one for all links or lengths randomly generated by a uniform distribution in (0,1000). Two types of graphs were considered. First, graphs corresponding to a grid of $n = p \times p$ nodes were created according to the following specifications (Figure 11):

(i) a maximum of 3 links are incident to each corner node;

(ii) a maximum of 5 links are incident to each non-corner frontier node;

(iii) a maximum of 8 links are incident to "interior" nodes.

The width of the grid varied from $p = 4$ to $p = 15$ (16 to 225 nodes).

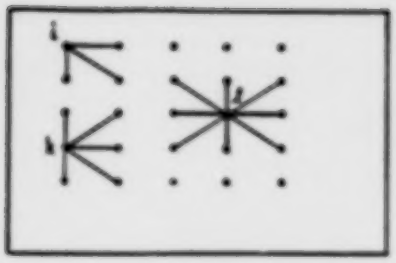

Figure 11: i a corner node, ℓ an interior node and
k a non-corner frontier node

These graphs are called pseudo-manhattan graphs. Diagonal links are not encountered in pure manhattan graphs.

Second, graphs with n nodes were constructed where each node has up to $(n-1)$ adjacent nodes. We call these last graphs "general" graphs. In our tests $n = 10, 20, ..., 100$.

For each value of p or n, according to the type of graph, 10 graphs with increasing total number of links and increasing number of arcs were generated. Note that in each graph a tree of edges was inserted to assure strong connectivity. Table 1 gives the appropriate data associated with the graphs generated.

Table 1.

Pseudo-Manhattan graph					General graph				
Graph No.	#nodes	#arcs	#edges	#links	Graph No.	#nodes	#arcs	#edges	#links
40	16	2	15	17	10	10	2	16	18
41	16	3	26	29	11	10	3	29	32
42	16	4	38	42	12	10	5	40	45
43	16	9	16	25	13	10	9	18	27
44	16	13	25	38	14	10	14	26	40
45	16	18	32	50	15	10	19	35	54
46	16	35	15	50	16	10	38	16	54
47	16	39	16	55	17	10	41	17	58
48	16	41	18	59	18	10	44	19	63
49	16	47	16	63	19	10	51	17	68
50	25	3	26	29	20	20	8	68	76
51	25	5	45	50	21	20	13	120	133
52	25	7	65	72	22	20	19	171	190
53	25	15	28	43	23	20	40	74	114
54	25	23	42	65	24	20	60	111	171
55	25	30	56	86	25	20	80	148	228
56	25	60	26	86	26	20	160	68	228
57	25	66	28	94	27	20	173	74	247
58	25	71	30	101	28	20	186	80	266
59	25	81	27	108	29	20	214	71	285

Table 1. (Continued)

Pseudo-Manhattan graph					General graph				
Graph No.	#nodes	#arcs	#edges	#links	Graph No.	#nodes	#arcs	#edges	#links
60	36	4	40	44	30	30	17	157	174
61	36	8	69	77	31	30	31	274	305
62	36	11	99	110	32	30	44	391	435
63	36	23	43	66	33	30	91	170	261
64	36	35	64	99	34	30	137	254	391
65	36	46	86	132	35	30	183	339	522
66	36	92	40	132	36	30	365	157	522
67	36	100	43	143	37	30	396	169	565
68	36	108	46	154	38	30	426	183	609
69	36	124	41	165	39	30	490	163	653
70	49	6	56	62	40	40	31	281	312
71	49	11	98	109	41	40	55	491	546
72	49	16	140	156	42	40	78	702	780
73	49	33	61	94	43	40	164	304	468
74	49	49	91	140	44	40	246	456	702
75	49	65	122	187	45	40	328	608	936
76	49	131	56	187	46	40	655	281	936
77	49	142	61	203	47	40	710	304	1014
78	49	153	65	218	48	40	764	328	1092
79	49	176	58	234	49	40	878	292	1170
80	64	8	76	84	50	50	49	441	490
81	64	15	132	147	51	50	86	772	858
82	64	21	189	210	52	50	123	1102	1225
83	64	44	82	126	53	50	257	478	735
84	64	66	123	189	54	50	386	717	1103
85	64	88	164	252	55	50	515	955	1470
86	64	176	76	252	56	50	1029	441	1470
87	64	191	82	273	57	50	1115	478	1593
88	64	206	88	294	58	50	1201	514	1715
89	64	236	79	315	59	50	1379	459	1838

Table 1. (Continued)

Pseudo-Manhattan graph					General graph				
Graph No.	#nodes	#arcs	#edges	#links	Graph No.	#nodes	#arcs	#edges	#links
90	81	11	98	109	60	60	71	637	708
91	81	19	171	190	61	60	124	1115	1239
92	81	27	245	272	62	60	177	1593	1770
93	81	57	106	163	63	60	372	690	1062
94	81	86	159	245	64	60	558	1035	1593
95	81	114	212	326	65	60	743	1381	2124
96	81	228	98	326	66	60	1487	637	2124
97	81	248	106	354	67	60	1611	690	2301
98	81	267	114	381	68	60	1735	743	2478
99	81	306	102	408	69	60	1991	664	2655
100	100	14	123	137	70	70	97	869	966
101	100	24	215	239	71	70	169	1522	1691
102	100	34	308	342	72	70	242	2173	2415
103	100	72	133	205	73	70	507	942	1449
104	100	108	200	308	74	70	761	1413	2174
105	100	144	266	410	75	70	1014	1884	2898
106	100	287	123	410	76	70	2029	869	2898
107	100	312	133	445	77	70	2198	942	3140
108	100	335	144	479	78	70	2367	1014	3381
109	100	385	128	513	79	70	2717	906	3623
110	121	17	151	168	80	80	126	1138	1264
111	121	29	265	294	81	80	221	1991	2212
112	121	42	378	420	82	80	316	2844	3160
113	121	88	164	252	83	80	664	1232	1896
114	121	132	246	378	84	80	995	1849	2844
115	121	176	328	504	85	80	1327	2465	3792
116	121	353	151	504	86	80	2654	1138	3792
117	121	382	164	546	87	80	2876	1232	4108
118	121	412	176	588	88	80	3097	1327	4424
119	121	473	157	630	89	80	3555	1185	4740

Table 1. (Continued)

Pseudo-Manhattan graph					General graph				
Graph No.	#nodes	#arcs	#edges	#links	Graph No.	#nodes	#arcs	#edges	#links
120	144	20	182	202	90	90	160	1442	1602
121	144	35	319	354	91	90	280	2524	2804
122	144	51	455	506	92	90	401	3604	4005
123	144	106	198	304	93	90	841	1562	2403
124	144	159	296	455	94	90	1261	2343	3604
125	144	212	395	607	95	90	1682	3124	4806
126	144	425	182	607	96	90	3364	1442	4806
127	144	461	197	658	97	90	3644	1562	5206
128	144	496	212	708	98	90	3925	1682	5607
129	144	569	190	759	99	90	4506	1502	6008
130	169	24	216	240	100	100	198	1782	1980
131	169	42	378	420	101	100	347	3118	3465
132	169	60	540	600	102	100	495	4455	4950
133	169	126	234	360	103	100	1040	1930	2970
134	169	189	351	540	104	100	1559	2896	4455
135	169	252	468	720	105	100	2079	3861	5940
136	169	504	216	720	106	100	4058	1782	5940
137	169	546	234	780	107	100	4505	1930	6435
138	169	588	252	840	108	100	4851	2079	6930
139	169	675	225	900	109	100	5569	1856	7425

Table 1. (Continued)

Pseudo-Manhattan graph				
Graph No.	#nodes	#arcs	#edges	#links
140	196	28	253	281
141	196	49	442	491
142	196	70	632	702
143	196	147	274	421
144	196	221	411	632
145	196	295	547	842
146	196	589	253	842
147	196	639	274	913
148	196	688	295	983
149	196	790	263	1053
150	225	33	292	325
151	225	57	511	568
152	225	81	731	812
153	225	170	317	487
154	225	256	475	731
155	225	341	633	974
156	225	682	292	974
157	225	739	317	1056
158	225	796	341	1137
159	225	914	304	1218

The simplex subroutine of the Land and Powell code [10] was used to solve the relaxations P_k. The computer used was CDC Cyber 855. Tables 2, 3, 4, and 5 give detailed statistics on the performance of our algorithm when applied to the graphs described in Table 1. The meaning of the columns of these tables is given below:

column 1:	graph number
column 2:	u_S for S being a most unbalanced set
column 3:	number of most unbalanced sets found in the initial graph
column 4:	number of balanced set constraints generated after the explosion procedure in the initial relaxation
column 5:	number of blossom inequalities generated in the initial relaxation
column 6:	total number of balanced set constraints present in the last relaxation
column 7:	total number of blossom inequalities present in the last relaxation
column 8:	total number of Gomory cuts present in the last relaxation
column 9:	total number of constraints in the last relaxation
column 10:	number of relaxations solved
column 11:	total number of simplex pivots
column 12:	execution time in seconds

Some comments are in order here. Columns 2 to 5 are pertinent to the initial relaxation. In a certain way they evaluate how far the initial graph is from being eulerian because of violations of the balanced sets condition (columns 2, 3 and 4) and of the evenness condition (column 5). Note about column 5 that all the blossom inequalities counted in this column are of the form β_S with $S = \{i\}$, node i being an odd node. In any graph, the number of odd nodes is even. It may happen however that the number of blossom inequalities given in column 5 is odd when some odd nodes are also unbalanced. In this case we prefer to generate a balanced set constraint. Columns 6 to 9 relate to the constraints contained in the last relaxation. Column 10 gives the number of relaxations that were necessary to solve in order to achieve optimality and column 11 gives the total number of simplex iterations that have been

computed when solving these relaxations. The meaning of column 12 is explicit.

Table 2. Pseudo-Manhattan Graph, $c_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
40				8		6		14	4	12	.9
41				12		2		14	2	14	.7
42				12		4		16	2	15	.8
43	1	4		11	1	2		14	2	12	.7
44				6		4		10	2	10	.6
45				8		6		14	3	14	1.0
46	2	4	4	8	4			12	1	8	.6
47	3	24	10	4	13	4		21	3	15	1.2
48	3	4	3	7	9	6		22	6	27	1.8
49	1	4	3	7	6	4		17	3	18	1.2
50				14		18		32	6	28	2.1
51				10		8		18	3	16	1.4
52				18		6		24	3	25	1.9
53	4	2	2	8	7	4		19	2	14	1.3
54	1	1	1	14	1	10		25	4	27	2.1
55				12				12	1	10	.8
56	4	4	2	6	7	8		21	4	20	1.8
57	6	8	9	13	16	7		36	4	38	3.0
58				16	1	6		23	2	21	1.4
59	4	1	1	13	3	4		20	3	20	1.7

Table 2. (Continued) Pseudo-Manhattan Graph, $c_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
60	1	2	2	18	2	16		36	6	41	3.5
61				20		16		36	8	34	5.1
62				24		16		40	6	39	5.5
63	2	70	27	15	30	18		63	9	53	6.3
64				18		24		42	12	41	7.1
65				16		12		28	4	28	3.2
66	8	128	33	16	45	16		77	8	60	10.7
67	8	12	13	15	23	12		50	5	42	5.9
68	2	72	22	14	25	10		49	3	28	5.3
69	2	3	2	22	4	4		30	2	25	2.6
70				22		20		42	5	40	3.9
71				24	1	26		51	9	54	8.9
72				30		20		50	8	49	10.8
73	5	18	8	27	11	20		58	6	63	6.8
74				22	1	10		33	3	31	4.2
75				24		18		42	8	43	9.1
76	7	56	22	18	35	17		70	7	61	10.8
77	10	75	50	17	84	27		128	19	215	51.4
78	3	42	28	22	33	30		85	13	197	27.5
79	5	16	4	20	8	16		44	5	53	7.3
80				30	1	28		59	10	57	13
81				34		24		58	9	54	17
82				44		28		72	8	64	19
83	7	4	1	36	13	24		73	9	90	16
84				32	1	20		53	8	47	14
85				42		20		62	9	59	19
86	15	125	50	30	79	21		130	13	143	47
87	12	65	50	31	83	30		144	17	234	70
89	6	44	26	32	47	46		125	12	230	62

Table 2. (Continued) Pseudo-Manhattan Graph, $a_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
90	2	2	1	35	4	20		59	4	54	10
91				32		20		52	3	45	10
92				40		68		108	27	117	76
93	8	144	8	37	26	49		112	15	162	44
94				40		36		76	9	70	27
95				44		48		92	13	96	44
96	23	64	26	32	79	50		161	27	527	176
97	21	48	50	40	81	30	1	152	19	248	101
98	8	40	29	30	46	32	4	112	14	188	69
100	2	108	12	42	11	42		95	10	95	32
101				46		64		110	21	116	69
102				42		52	3	97	26	98	102
103	4	48	12	51	15	26		92	5	81	24
104	1	18	6	58	8	46	3	115	12	146	76
105				54		26		80	6	74	31
106	24	2465	50	43	125	54	1	223	28	547	433
110	1	160	30	55	29	64	24	172	43	239	179
111				52		54		106	11	100	61
112				58		96		154	26	163	146
114	3	4	4	58	9	42		109	6	111	48
115				58		38		96	8	95	53
118	4	160	33	46	32	58		136	12	172	152
120	3	455	50	62	58	136		256	35	339	279
121				72		88	2	162	28	184	191
122				74		40		114	7	103	64
124	1	1	1	74	1	66		141	16	138	126
125				76		36		112	8	106	70

Table 2. (Continued) Pseudo-Manhattan Graph, $c_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
131				80		80		160	12	154	128
132				84		86		170	16	176	170
134	1	3	3	82	7	60		149	11	156	133
135				90		68		158	10	154	126
140	3	1203	50	96	52	108		256	16	297	384
141				100		128		228	32	232	426
144	1	2	2	104	9	112		225	22	250	391
145				106		84		190	12	188	194
151				114		82		196	12	183	244
154	2	3	2	112	4	126		242	22	261	498

Table 3. Pseudo-Manhattan Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
40				8	1	8		17	4	16	1.0
41				12				12	1	13	.5
42				12				12	1	18	.6
43	1	4		11	5	4		20	6	23	1.4
44				6				6	1	5	.4
45				8		6		14	3	13	1.0
46	2	4	4	8	4			12	1	11	.6
47	3	24	10	4	9	4		17	3	16	1.2
48	3	4	3	7	6	6	2	21	7	27	2.0
49	1	4	3	7	4			11	1	5	.5
50				14		16		30	5	26	2.0
51				10		12		22	4	24	1.9
52				18				18	1	26	1.1
53	4	2	2	8	9	7		24	3	17	1.4
54	1	1	1	14	1	6		21	4	23	2.0
55				12				12	1	14	.9
56	4	4	2	6	8	8		22	6	33	2.7
57	6	8	9	13	14	6		33	4	41	3.0
58				16	1	2		19	2	25	1.5
60	1	2	2	18	3	16		37	5	40	3.2
61				20				20	1	26	1.4
62				24		2		26	2	33	2.6
63	2	70	27	15	29	16		60	8	52	6.1
64				18				18	1	24	1.5
65				16		16		32	7	41	5.3
66	8	128	33	16	42	18	2	78	14	123	17.4
67	8	12	13	15	22	8		45	6	56	7.0
68	2	72	22	14	23	8		45	5	42	6.7
69	2	3	2	22	6	12	48	88	57	298	62.0

Table 3. (Continued) Pseudo-Manhattan Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
70				2	1	26		29	7	45	5.0
72				30		2		32	2	41	4.2
73	5	18	8	27	14	18	1	60	11	79	11.3
74				22		4		26	3	35	4.1
75				24		16		40	9	46	10.1
76	7	56	22	18	30	3		51	5	79	10.4
78	3	42	28	22	21	12		55	7	49	10.2
79	5	16	4	20	25	6	2	53	21	84	20.3
80				30	1	34		65	12	78	16
81				34		24		58	11	75	22
82				44		6		50	4	64	12
83	7	4	1	36	9	16		61	4	78	10
84				32		14		46	8	48	13
85				42		12		54	7	68	16
87	12	65	50	31	75	32		138	16	511	121
88	4	54	28	31	37	4	6	78	13	127	36
89	6	44	26	33	31	6		70	4	116	24
90	2	2	1	35	2	18		55	4	52	10
91				32		26		58	8	63	20
92				40		14		54	8	62	23
93	8	144	8	37	22	30		89	14	179	50
94				40	4	34	5	83	23	189	77
95				44		4		48	3	60	13
98	8	40	29	30	37	22	1	90	10	134	45
99	3	144	50	40	70	6	3	119	18	231	93
100	2	108	12	42	8	48		98	15	98	43
101				46		34		80	8	84	29
102				42		20		62	6	69	26
103	4	48	12	51	21	44		116	10	143	45
104	1	18	6	58	8	32	15	113	28	192	153
105				54		34	2	90	17	122	73

Table 3. (Continued) Pseudo-Manhattan Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
110	1	160	30	55	23	52		130	10	119	47
111				52		68		120	21	163	129
112				58		34		92	13	108	73
114	3	4	4	58	6	32		96	8	113	54
115				58				58	1	74	16
118	4	160	33	46	23	36		105	19	173	173
120	3	455	50	62	46	79		187	11	186	98
121				72		46	3	121	19	153	126
122				74		38	7	119	18	171	194
124	1	1	1	74	1	42		117	18	151	126
125				76		12		88	7	117	63
131				80		44		124	19	130	177
132				84		28		112	11	136	113
140	3	1203	50	96	47	58		201	7	164	235
142				86		6		92	4	103	69
145				106		28		134	7	196	126
151				114		52		166	16	214	303
152				92		60		152	26	228	477
155				120		24		144	10	188	262

Table 4. General Graph, $\alpha_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
10				4		2		6	2	5	0.4
11				4		2		6	2	6	0.5
12				6		4		10	3	8	0.8
13	1	4	2	6	3	2		11	2	9	0.6
14				2				2	1	3	0.3
15				6				6	1	5	0.1
16	2	16	6	1	14			15	3	10	0.7
17	6	1		4	2	4		10	3	11	0.7
18				8	1	2		11	2	9	0.7
19	1	1		5	2	6		13	4	13	1.1
20				10		6		16	3	19	1.6
21				10		4		14	2	12	2.0
22				10		6		16	3	16	5.0
23	1	2	1	10	4	4		18	4	20	1.7
24				10		8		18	4	17	2.9
25				10		4		14	3	16	4.2
26	3	8	6	6	10	4		20	3	13	1.3
27	3	3	2	8	6	4		18	4	21	2.8
28				14		4		18	3	13	3.2
29				8		18		26	10	25	6.3
30				16		12		28	5	28	6.2
31				14		8		22	4	19	14.4
32				12		6		18	3	18	24.4
33	4	1	1	21	2	8		31	3	28	3.5
34				18		8		26	5	23	11.9
35				12		16		28	6	29	30.1
36	11	1	1	11	20	16		47	12	73	9.7
37				16		10		26	6	26	8.5
38				16		6		22	4	23	12.4
39				16		16		32	7	27	16.2

Table 4. (Continued) General Graph, $c_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
40				20		10		30	4	26	12.5
41				18		8		26	2	22	26.1
42				20		8		28	2	22	66.6
43	1	1		21	1	8		30	3	30	7.1
44				20		20		40	7	37	46.5
45				18		18		36	6	42	101.5
46	14	2	2	17	15	12		44	7	48	10.6
47	3	2	1	19	2	14		35	6	32	22.7
48				12		6		18	3	19	25.8
49				24		10		34	4	31	28.6
50				26		14		40	6	40	37.3
51				28		14		42	3	40	96.8
52				20		22		42	6	41	433.6
53				28		12		40	5	40	20.2
54				26		26		52	10	47	151.9
55				26		8		34	4	32	188.7
56	14	15	15	23	29	14		66	7	58	23.8
57	2	1		22	2	20		44	5	42	42.3
58				22		20		42	6	46	122.1
59				26		12		38	4	37	64.8
60				32		26		58	7	65	85
61				30		20		50	5	52	312
63	1	1		33	1	18		52	7	55	56
64				32		16		48	3	49	121
65				22		14		36	5	37	488
66	15	50	50	22	76	39		137	18	138	126
67				26		22		48	5	48	84
68				28		24		52	7	53	298
69				28		20		48	6	53	184

Table 4. (Continued) General Graph, $c_{ij} \geq 1$

1	2	3	4	5	6	7	8	9	10	11	12
70				30		24		54	7	47	159
73				30		12		42	2	36	37
74				34		14		48	3	42	219
76	13	48	48	26	76	54		156	29	259	282
77	1	1		39	1	22		62	5	56	175
80				36		30		66	7	69	276
83				34		28		62	6	61	130
84				42		12		54	4	46	490
87				42		24		66	7	59	333
90				40		26		66	4	65	308
93				42		28		70	6	62	197
96	35	9	9	35	46	54	2	137	29	307	452
103				46		36		82	6	79	287
106	17	1	1	46	16	42		104	13	100	195

Table 5. General Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
10				4				4	1	5	0.4
11				4				4	1	6	0.5
12				6				6	1	8	0.6
13	1	4	2	6	1			7	1	9	0.5
14				2				2	1	3	0.4
15				6				6	1	8	0.6
16	2	16	6	1	14			15	3	10	1.0
17	6	1		4	2	4		10	3	10	1.0
18				8				8	1	11	0.7
19	1	1		5	1			6	1	8	0.6
20				10		2		12	2	15	1.4
21				10				10	1	17	1.6
22				10		2		12	2	15	4.0
23	1	2	1	10	1			11	1	13	1.0
24				10				10	1	14	1.5
25				10				10	1	15	2.5
26	3	8	6	6	9	2		17	2	12	1.4
27	3	3	2	8	4			12	1	15	1.6
28				14				14	1	24	2.6
29				8				8	1	11	1.6
30				16				16	1	19	2.6
31				14				14	1	19	6.5
32				12				12	1	17	13.6
33	4	1	1	21	3	12		36	6	52	7.4
34				18				18	1	27	5.1
35				12				12	1	18	9.4
36	11	1	1	11	17	5		33	8	36	7.0
37				16				16	1	25	3.7
38				16		8		24	5	34	16.3
39				16				16	1	20	5.5

Table 5. (Continued) General Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
40				20				20	1	35	6.2
41				18				18	1	39	18.7
42				20		4		24	3	35	86.5
43	1	1		21	1			22	1	38	4.8
44				20				20	1	35	13.8
45				18				18	1	36	32.3
46	14	2	2	17	16	12		45	9	71	15.4
47	3	2	1	19	2			21	1	36	9.7
48				12		6		18	4	23	32.9
49				24				24	1	52	18.9
50				26				26	1	38	12.5
51				28				28	1	46	50.3
52				20				20	1	38	125.2
53				28				28	1	49	8.8
54				26				26	1	59	33.8
55				26				26	1	59	84.9
56	14	15	15	23	27	20		70	10	152	42.2
57	2	1		22	1			23	1	43	18.4
58				22		10		32	6	46	120.6
59				26				26	1	38	32.4
60				32		8	4	44	9	73	113
61				30				30	1	58	110
63	1	1		33	1			34	1	82	20
64				32				32	1	71	69
65				22				22	1	35	164
67				26				26	1	55	36
69				28		14		42	8	67	224

Table 5. (Continued) General Graph, $c_{ij} = 1$

1	2	3	4	5	6	7	8	9	10	11	12
70				30				30	1	49	44
73				30				30	1	72	28
74				34				34	1	70	120
75				36				36	1	70	373
76	13	48	48	26	58	12	6	102	13	229	150
77	1	1		39	1			40	1	71	75
78				40				40	1	62	176
79				34				34	1	63	124
80				36				36	1	71	78
83				34				34	1	61	42
84				42		4		46	3	97	407
87				42				42	1	77	111
93				42				42	1	104	70
94				52				52	1	121	404
100				48				48	1	111	208
103				46		12		58	7	111	334
106	17	1	1	46	16			62	7	158	146
107				52		2		54	2	127	400

The preceding tables contain results about only the problems that were solved successfully. In most unsolved problems, the main reason of failure is that the maximum allowable time of 500 seconds has been attained. Problems that were not solved within this time limit were also unsolved when we tried to increase the time limit. In some problems also the memory available was not sufficient to store all the constraints generated. These two causes of failure are related in most cases to the same phenomenon: difficulty in achieving integrality where most of the time is spent on generating Gomory cuts.

The largest problem solved in terms of nodes contains 225 nodes (341 arcs, 633 edges). In terms of links, the largest problem solved has 6,435 links amongst which are 4,505 arcs and 1,930 edges. To our knowledge, these are the largest MCPP ever solved. Globally, our relaxation process performed amazingly well. Very few constraints are generated relative to the exponential number of constraints in the whole model and the execution times are quite good relative to the complexity of the MCPP. Note here that no particular efforts have been made to refine the computer code. The LP code of Land and Powell is good but rather old and our flow algorithm is not the most sophisticated. Nevertheless, our execution times are very low mainly due to the sharp quality of our relaxations. It is largely accepted today that good models are more important than good programming when solving difficult combinatorial problems.

If we look at our results in more detail, we can make the following observations:

(i) Independent of problem type and of cost structure, the mean number of constraints generated for successful problems is approximately equal to the number of nodes. This is a clear indication of the quality of our relaxations and this also explains the low figures observed in column 10.

(ii) The mean number of blossom inequalities inserted in the initial relaxation is about half the number of nodes for pseudo-manhattan graphs as well as for general graphs. This is due to the fact that, in the graphs generated, about half the nodes were odd.

(iii) The more the original graph is unbalanced (column 2), the more balanced set constraints are put in the initial relaxation in every table. The reason is that

most unbalanced sets are more numerous when the graph is badly unbalanced. Frequently, the number of constraints generated by the explosion procedure (column 4) is less than the number of most unbalanced sets found (column 3). The reason is that many unbalanced sets are constituted of a "basic" most unbalanced set to which are added balanced nodes or balanced sets. The explosion procedure isolates these balanced nodes or sets and does not generate any constraints from them just because they are balanced.

(iv) After the initial relaxation, balanced set constraints generated are more numerous than blossom inequalities when the original graph is greatly unbalanced. The converse is true if the graph is not unbalanced or a little unbalanced. Again here, the explanation is simple: by balancing sets we not only obtain an equal number of leaving and entering arcs but also we make these sets even. So the more often balanced set constraints are generated the less often blossom inequalities are needed to make the whole graph even.

(v) For all the problems that were solved, very few Gomory cuts were needed. The converse is true for unsolved problems. It seems that with cutting planes, integrality is attained very fast or not at all. Note that in an earlier version of our algorithm we tried to achieve integrality first, step 4 becoming step 2. The results were bad and too many Gomory cuts were generated. We notice also that, in many problems, the introduction of only one blossom inequality restores immediately the integrality of the variables so that Gomory cuts were not needed.

(vi) One of the most surprising results of our method is that it was never necessary

to generate a v_i variable. We do not have yet a theoretical explanation of this. We tested another version of the algorithm where all the v_i variables as well as the associated constraints were always inserted in the initial relaxation. The results were good but not as good as the results presented here.

(vii) It is difficult to compare the execution times between pseudo-manhattan graphs and general graphs because their respective number of links is so much different. Pseudo-manhattan graphs with $16=4\times4$ nodes and general graphs with 10 nodes have about the same number of links and the execution times are about the same regardless of the cost structure. General graphs are much more dense than pseudo-manhattan graphs having approximately the same number of nodes (except for $n=4\times4$ and $n=10$). That is probably why the execution times observed when solving general graphs is greater than when solving pseudo-manhattan graphs.

(viii) The execution times for pseudo-manhattan graphs with $c_{ij} \geq 1$ are greater in general than those observed when $c_{ij} = 1$. The same observation can be made for general graphs.

(ix) The more links that are present in the original graph and the more it is unbalanced, the greater are the execution times.

We will now compare our method with the other optimal algorithms that we have found in the literature. First, the algorithm of Christofides et al. [3], devised in 1983, enabled the authors to solve MCPP's involving up to 50 nodes, 66 arcs and 39 edges. It took about 500 seconds on a UNIVAC 1100/60 computer to solve the largest

problem. The execution times seem quite high relative to the small number of arcs and edges present in their problems. The Christofides et al.'s algorithm is a branch-and-bound algorithm using two bounds derived from two different Lagrangean relaxations. The authors used a mathematical model for the MCPP where two variables are associated to each edge of the graph, one for each direction in which an edge can be traversed. Their model contains also the equivalent of our v_i variables. So their model contains many more variables than ours. The main drawback of Christofides et al.'s approach seems to be that integrality on the variables is difficult to attain. We think that their method is inferior to ours.

Grötschel and Win in 1988 [7] elaborated an optimal cutting plane algorithm for the Windy Postman Problem (WPP). By replacing an arc (i, j) of cost c_{ij} by an edge (i, j) of costs c_{ij} and ∞, the MCPP can be transformed into a WPP. Grötschel and Win solved successfully with their method 8 MCPP's constructed from 6 real world transportation networks. The largest graph contains 172 nodes, 131 edges and 116 arcs and the associated MCPP was solved in about 74 seconds on a NORSK DATA ND-540 computer. Here again the underlying model of the Grötschel and Win approach associates two variables to each edge of the graph. Their results with cutting planes to achieve integrality seem better than those obtained by Christofides et al. It is difficult to compare our method with the one of Grötschel and Win because of the difference in the computer used. All we can say for sure is that, with our method, we succeeded in solving much larger problems. Moreover, the total number of constraints generated by the Grötschel and Win algorithm for problems of similar

size is much higher than the total number of constraints generated by our method.

6. Conclusion

We have presented in this paper a new model and a new optimal algorithm for the mixed Chinese postman problem. We also have devised a new and very efficient way to check if a given mixed graph is balanced using quadratic programming. The extensive computer results given in this paper show clearly that our approach is quite successful and very large MCPP's were solved. One of the main advantages of our algorithm lies in its flexibility. It can be easily adapted to the case where the original graph is strictly oriented, strictly non-oriented or even when the original graph is a multigraph. Moreover, lower and/or upper bounds on the variables can be considered with minor modifications. We have already used our method on real networks as a part of a larger algorithm to construct routes for snow removal in the city of Montreal. We are now modifying our approach in order to solve the rural postman problem. Moreover, a new optimal branch-and-bound algorithm and a new heuristic are currently under study for the MCPP.

7. Acknowledgements

First, we would like to thank Professor B.T. Smith whose helpful suggestions and comments have greatly improved the quality of the paper. Second, this work would not have been possible without the great computer skills of Patrick Samson who devised the extensive computer code of our algorithm. We truly are grateful to him.

8. References

[1] Bellmore, M., Hong, S., "Transformation of Multisalesmen Problem to the Standard Traveling Salesman Problem", *Journal of the ACM* 21, 500–504 (1974).

[2] Bodin, L.D., Kursh, S.J., "A Computer-Assisted System for the Routing and Scheduling of Street Sweepers", *Operations Research* 26, 525–537 (1978).

[3] Christofides, N., Benavent, E., Campos, V., Corberan, A., Mota, E., "An Optimal Algorithm for the Mixed Chinese Postman Problem", Philips & Klingman Eds., Network Algorithms (Special issue of *Discrete Applied Mathematics*) (1986).

[4] Edmonds, J., Johnson, E.L., "Matching, Euler Tours and the Chinese Postman", *Mathematical Programming* 5, 88–124 (1973).

[5] Ford, L.R., Fulkerson, D.R., *Flows in Networks*, Princeton University Press (1962).

[6] Frederickson, G.N., Hecht, M.S., Kim, C.E., "Approximation Algorithm for Some Routing Problems", *SIAM Journal of Computing* 7, 178–193 (1978).

[7] Grötschel, M., Win, Z., "A Cutting Plane Algorithm for the Windy Postman Problem", Report No. 76, Institut für Mathematik, Universität Augsburg (1988).

[8] Jorker, R., Volgenant, T., "Transforming Asymmetric into Symmetric Traveling Salesman Problem", *Operations Research Letters* 2, 161–163 (1983).

[9] Kapauf, C.H., Koehler, G.J., "The Mixed Postman Problem", *Discrete Applied Mathematics* 1, 89–103 (1979).

[10] Land, A.H., Powell, S., *Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete*, Wiley (1973).

[11] Minieka, E., "The Chinese Postman Problem for Mixed Networks", *Management Science* 25, 643–648 (1979).

[12] Nobert, Y., "Construction d'algorithmes optimaux pour des extensions au problème du voyageur de commerce", Ph.D. thesis, Université de Montréal (1983).

[13] Papadimitriou, C.H., "On the Complexity of Edge Traversing", *Journal of the Association for Computing Machinery* 23, 544–554 (1976).

[14] Pearn, W.L., Assad, A., Golden, B., "Transforming Arc Routing into Node Routing Problems", *Computers & Operations Research* 4, 285–288 (1987).

[15] Picard, J.C., Queyranne, M., "On the Structure of All Minimum Cuts in a Network and Applications", *Mathematical Programming* 13 (1980).

[16] Picard, J.C., Ratliff, H.D., "Minimum Cuts and Related Problems", *Networks* 5, 357–370 (1975).